Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Automating Enterprise Onboarding with PowerShell and Chef: A Scalable Framework for PLM Environments

Swami Venkatesh Mandepu

Independent Researcher, USA

ARTICLE INFO

Received: 11 March 2026

Accepted: 15 March 2026

ABSTRACT

Enterprise onboarding processes in Product Lifecycle Management environments remain plagued by manual procedures, inconsistent execution, and resource-intensive operations that create substantial delays and configuration errors. Traditional onboarding workflows require extensive IT administrator involvement for Active Directory user provisioning, application role assignments, and system configurations, particularly in manufacturing and engineering organizations where specialized PLM applications like Teamcenter and UGNX demand precise role-based access controls and environment-specific configurations. This technical review presents a scalable automation framework that integrates Windows PowerShell scripting for identity and directory management with Chef-based configuration management for PLM applications, establishing an Infrastructure-as-Code paradigm that treats user provisioning and application configuration as version-controlled, repeatable processes. The framework implements a three-tier architecture comprising orchestration, execution, and validation layers that coordinate identity management operations with application configuration tasks through parallel processing, significantly reducing onboarding completion times while minimizing configuration errors. Implementation at a global aerospace manufacturing organization demonstrates substantial operational improvements, including reduced onboarding durations, decreased error rates, and enhanced compliance with enterprise security policies. The framework's extensibility enables expansion beyond initial engineering onboarding to encompass manufacturing execution systems, quality management applications, and enterprise resource planning tools, providing a foundation for comprehensive enterprise automation initiatives that modernize IT operations while respecting constraints of legacy application ecosystems.

Keywords: Infrastructure-as-Code, Enterprise Onboarding Automation, Product Lifecycle Management, PowerShell Scripting, Chef Configuration Management

1. Introduction

Enterprise onboarding procedures, especially when tied to Product Lifecycle Management (PLM) systems, tend to be manual, ad-hoc, and require significant direction from IT. Active Directory (AD) user provisioning, application-specific role assignments, and configuration of systems demand considerable IT resources, and human error persists throughout this process. Traditional manual onboarding procedures in large enterprises consume substantial IT administrator time per employee, with engineering roles requiring specialized PLM access typically demanding considerably more time due to complex application configurations and multi-system integration requirements. Active Directory Domain Services operates as a hierarchical database structure that stores all information related to network objects such as user accounts, computers, groups, and organizational units, thus allowing centralized authentication and authorization across enterprise and Windows environments

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

through LDAP implementations and Kerberos-based service security mechanisms [1]. Challenges increase due to modern enterprise environments that function across multiple geographies with multiple physical sites, which provide complications above and beyond maintaining a standard onboarding procedure where manual approaches simply do not scale in particularly when demonstrating compliance with security policies or regulations. Organizations that operate across multiple continents with distributed activities demonstrate low consistency rates with onboarding new roles unless manual procedures are followed; configuration differences can result in 50% or more of engineering professionals in their first week reporting access issues to systems and blogs or complaints about applications not properly functioning.

The challenge is compounded in manufacturing and engineering organizations referring to more specialized PLM applications, for example, Teamcenter and UGNX, that require specific configurations related to access control to fulfill role-based functionality and environment configurations that vary among development, testing, and production systems. Teamcenter deployments in aerospace and automotive enterprises typically manage extensive engineering documents with access permissions spanning numerous distinct user roles, each requiring specific security group memberships, organizational unit placements, and application-level privilege assignments. UGNX client installations demand configuration of numerous environment variables, registry entries spanning multiple registry hives, and integration parameters connecting to translation servers, license managers, and PLM data vaults. Manual configuration of these parameters across diverse workstation hardware configurations results in significant error rates, with misconfigurations frequently requiring substantial troubleshooting and remediation effort per incident.

This paper presents a scalable automation framework that integrates Windows PowerShell scripting for identity and application management with Chef-based configuration management for PLM applications. The framework addresses critical gaps in traditional onboarding methodologies by establishing an Infrastructure-as-Code approach that treats user provisioning and application configuration as version-controlled, repeatable processes rather than manual procedures executed through graphical interfaces. Chef Infra operates as a declarative configuration management platform utilizing Ruby-based domain-specific language to define system states as code, with agent-server architecture enabling centralized policy distribution and enforcement across heterogeneous infrastructure environments, including physical servers, virtual machines, and cloud instances [2]. The solution architecture leverages PowerShell's native integration with Active Directory Services to automate user account creation, security group membership assignment, and organizational unit placement through scripted workflows executing discrete operations per onboarding event, while Chef recipes manage deployment and configuration of PLM client applications across distributed engineering workstations through resource definitions encompassing extensive configuration parameters.

The framework significantly reduces onboarding times from days to hours, minimizes configuration errors from baseline rates to substantially lower post-automation rates, and ensures compliance with enterprise security policies through programmatic enforcement of standardized configurations. A case study demonstrates how the solution enabled consistent global onboarding for engineering teams spanning multiple geographical locations, processing substantial onboarding events during the first operational year with high success rates and saving considerable engineering workdays annually.

2. Background and Related Work

The progress of automated onboarding for enterprises exemplifies sweeping changes in IT service management and DevOps thinking. Traditionally, the onboarding process in large organizations has involved a ticketing system whereby an IT Administrator delivers provisioning-based manual

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

processes through the use of documented step-by-step processes repeatedly. In a manually-based system, tickets to resolve standard onboarding requests typically take longer to complete, while provisioning for complex engineering roles can take much longer when dependencies across multiple systems necessitate sequential deliveries. Continuous delivery methods originated the practice of treating infrastructure and configuration as code-based artifacts, such that teams can version control, test, and deploy code through automation and the pipeline to eliminate duplicated effort and reduce the time for deploying from weeks to a few hours while producing a reliable and consistent result. The Infrastructure-as-Code paradigm gained significant traction with the emergence of configuration management tools, including Puppet, Chef, and Ansible, which provide declarative frameworks for defining desired system states and automatically enforcing those states across distributed infrastructure comprising thousands to tens of thousands of managed nodes. The 2024 State of DevOps Report demonstrates that organizations implementing platform engineering approaches with automated configuration management achieve faster software delivery velocity and experience fewer production incidents compared to organizations relying on manual infrastructure operations, while high-performing teams report lower change failure rates through systematic application of infrastructure automation practices [3]. Configuration management platforms enable organizations to reduce configuration drift incidents substantially compared to manual configuration approaches, while decreasing mean time to recovery from configuration-related failures through automated remediation capabilities.

Product Lifecycle Management systems present unique challenges for automation due to their complex client-server architectures and specialized configuration requirements. Teamcenter represents one of the most widely deployed PLM platforms in manufacturing and aerospace industries, supporting engineering data management for assemblies containing numerous individual components. The application requires extensive client-side configuration, including environment variables, registry settings spanning multiple registry hives, and integration with CAD tools such as UGNX, requiring bidirectional data translation mechanisms.

Chef configuration management provides a Ruby-based domain-specific language for defining infrastructure configurations as executable code. The Chef architecture implements a master-slave configuration where the Chef Server functions as the central repository storing cookbooks, recipes, policies, and metadata about managed infrastructure nodes, while Chef Workstation serves as the development environment where infrastructure engineers author and test configuration code before uploading to the server, and Chef Client agents execute on managed nodes to retrieve configuration policies and converge systems to desired states through idempotent resource application [4]. Research on Chef adoption patterns indicates that organizations achieve operational benefits through reduced configuration drift affecting minimal percentages of managed nodes, improved audit compliance enabling rapid configuration state reports, and faster recovery from system failures through automated rebuild procedures completing quickly versus substantially longer manual recovery times.

Category	Traditional Approach	Modern Automated Approach
Identity Management	Manual creation of user accounts, assigning roles and groups through GUI tools; high error rates and inconsistent execution.	Automated user provisioning using PowerShell scripting, standardized role assignment, policy-based group membership, and idempotent identity operations.
Application Configuration	Manual installation of PLM applications (Teamcenter, UGNX), editing configuration files, setting	Chef-based configuration management defining desired application states as code, automated

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

	environment variables, and updating registry entries on each workstation.	deployment across distributed workstations, and environment-specific configuration logic.
Process Consistency	Procedures vary across teams, geographies, and technicians; onboarding times depend on manual sequencing and availability of IT personnel.	Workflow orchestration via PowerShell controller and Chef automation ensures consistent, repeatable, version-controlled onboarding processes.
Scalability	Manual steps do not scale across multi-site global enterprises; significant delays during peak onboarding cycles.	Scalable pipeline supporting parallel execution of identity management and application configuration tasks, significantly reducing processing time.
Compliance & Auditability	Limited visibility into manual steps; inconsistent documentation and audit gaps.	Automated logging, validation checks, and InSpec compliance testing provide full traceability and continuous policy enforcement.

Table 1: Evolution of Enterprise Onboarding Automation Technologies [3,4]

3. Framework Architecture and Implementation

The proposed automation framework implements a three-tier architecture comprising orchestration, execution, and validation layers that work in concert to deliver consistent onboarding experiences across distributed enterprise environments. The orchestration layer serves as the framework's control plane, managing workflow sequences and coordinating between identity management operations and application configuration tasks through a state machine processing discrete workflow steps per onboarding event. This layer utilizes a centralized PowerShell-based controller that interprets onboarding requests submitted through ServiceNow integration or direct API calls, accepting JSON payloads containing structured parameters. The controller maintains state information in a SQL Server database implementing multi-layered security architecture that includes authentication mechanisms supporting Windows Authentication and SQL Server Authentication modes, authorization controls through role-based permissions and row-level security policies, encryption capabilities for data at rest using Transparent Data Encryption and data in motion through Transport Layer Security protocols, and auditing features capturing security-relevant events for compliance verification [5]. Request validation includes verification of manager approval chains traversing organizational hierarchies, department code accuracy cross-referenced against enterprise resource planning systems, and geographical location assignments determining specific configuration profiles.

The execution layer encompasses two parallel automation paths that operate synchronously to minimize total onboarding time through concurrent task processing. The identity management path leverages PowerShell scripts that interface with Teamcenter through the Organization module, executing operations beginning with user account creation in designated organizational units defined by the organization. The scripts apply standardized naming conventions combining first name, last name, and employee identifier components, set initial passwords following complexity requirements with mixed character classes, configure password policies including expiration dates and forced change-at-first-login flags, and assign users to appropriate security groups controlling access to network resources. Group membership assignment follows a role-based access control model where engineering roles automatically receive membership in PLM user groups, CAD designer groups, and

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

department-specific distribution lists. The PowerShell scripts implement idempotency principles through conditional logic checking for existing accounts before creation, which supports rerunning scripts after transient failures without creating duplicate accounts.

The application configuration path utilizes Chef cookbooks specifically designed for PLM application deployment, with the primary Teamcenter cookbook comprising Ruby code organized into recipes addressing distinct configuration domains. InSpec compliance testing framework enables infrastructure verification through declarative test specifications written in a domain-specific language that express security and compliance requirements as executable code, allowing automated validation of system configurations against organizational policies and regulatory standards through continuous assessment of infrastructure states [6]. The Teamcenter cookbook defines resources for installing the Teamcenter rich client application, configuring connection parameters to organization-specific environments, setting up local cache directories with appropriate size limits and cleanup policies, and establishing integration with UGNX through configuration files supporting distinct file format conversions. Chef recipes employ conditional logic to apply environment-specific configurations, with development engineers receiving connections to testing databases containing test engineering objects, while production engineers access manufacturing repositories managing production parts and assemblies. The recipes also manage license server configurations, updating license file paths and server hostnames based on geographical location to ensure connections to geographically proximate license servers that minimize network latency and improve application performance.

The validation layer implements automated testing procedures that verify the successful completion of all provisioning steps before marking an onboarding workflow as complete. PowerShell-based validation scripts query Teamcenter Organization to confirm user account existence, verify group memberships match role requirements by comparing actual memberships against expected sets defined in role matrices, and test authentication by attempting LDAP binds with generated credentials. Chef compliance profiles written using the InSpec framework validate application installations by checking for expected file system artifacts, verifying registry key values, testing network connectivity to PLM servers and license managers, and confirming that application launch succeeds without errors.

Architecture Layer	Primary Components	Key Capabilities
Orchestration Layer	PowerShell-based controller with ServiceNow integration and SQL Server database	Workflow sequence management, state tracking, request validation, and audit logging
Execution Layer	Parallel automation paths for identity management and application configuration	PowerShell scripts for Active Directory operations and Chef cookbooks for PLM deployment
Validation Layer	PowerShell validation scripts and InSpec compliance profiles	Automated testing procedures, authentication verification, and application installation validation
Security Integration	Multi-layered database security and privileged access management systems	Role-based permissions, encryption capabilities, and secure credential management

Table 2: Three-Tier Automation Framework Architecture [5, 6]

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4. Case Study and Results

The framework deployment case study examines implementation at a global aerospace manufacturing organization with a substantial employee population across engineering, manufacturing, and administrative functions distributed across facilities in North America, Europe, and Asia. Before automation implementation, the organization's IT operations team manually processed engineering onboarding requests that required multiple days to complete, primarily due to sequential dependencies between Active Directory provisioning, application installation scheduling, and configuration of department-specific PLM environments. The manual process involved IT technicians executing documented procedures that included creating user accounts through the Active Directory Users and Computers interface, installing Teamcenter and UGNX clients from network shares or physical media, manually editing configuration files to specify server connections, and coordinating with engineering managers to verify appropriate workspace access and project assignments. Error rates represented a significant portion of onboarding requests, manifesting as incorrect group memberships, misconfigured application connections, or incomplete installations that required rework and extended onboarding timelines.

The organization deployed the automation framework in a phased approach, beginning with pilot testing in a single engineering department before expanding to global deployment over an extended implementation period. The pilot phase focused on refining PowerShell scripts for Active Directory operations and developing initial Chef cookbooks for Teamcenter deployment, with iterative testing and validation against representative engineering workstation configurations. During pilot operations, the framework successfully onboarded new engineering employees with substantially reduced completion time from request submission to validated completion, representing a significant reduction in onboarding time compared to manual procedures. DevOps upskilling initiatives emphasize that mastering automation technologies requires structured learning paths encompassing infrastructure-as-code principles, configuration management tools, continuous integration and deployment practices, and collaborative workflows that bridge traditional operational boundaries between development and infrastructure teams [7]. Error rates during pilot operations dropped considerably, with all errors attributable to environmental issues such as network connectivity problems during Chef client runs rather than configuration mistakes or procedural omissions.

Full production deployment commenced following successful pilot completion, with the framework processing substantial onboarding requests for engineering employees during the initial operational period. Operational metrics demonstrated consistent performance improvements with an average onboarding completion time, maintaining the substantial time savings observed during pilot operations. The organization quantified direct labor savings by comparing estimated manual effort hours against automation execution time, calculating significant annual savings in engineering workdays. Building effective business cases for automation requires comprehensive analysis across five critical dimensions, including identifying specific operational pain points and inefficiencies in current processes, quantifying baseline performance metrics and cost structures, evaluating available automation technologies and implementation approaches, calculating return on investment through projected cost savings and productivity improvements, and establishing implementation roadmaps with defined milestones and success criteria [8].

Beyond quantitative time and labor savings, the organization realized several qualitative benefits from automation adoption. Standardization of onboarding procedures eliminated configuration inconsistencies that previously caused application performance issues or access control problems, resulting in fewer help desk tickets related to PLM connectivity and authentication failures. The audit logging capabilities of the framework documented onboarding events with a high level of detail, contributing to compliance standards for quality management systems and aerospace industry security requirements. Engineering managers stated that employees had better onboarding

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

experiences because new hires faced fewer technical constraints during their initial period and received standardized configurations irrespective of which geographic location they onboarded.

Deployment Phase	Implementation Activities	Operational Outcomes
Pilot Testing	PowerShell script refinement and initial Chef cookbook development in single engineering department	Successful onboarding with substantial time reduction and decreased error rates
Production Deployment	Expansion to global operations with enhanced cookbooks and scripts across all locations	Consistent performance improvements with maintained time savings and reduced errors
Operational Benefits	Standardization of onboarding procedures and comprehensive audit logging capabilities	Eliminated configuration inconsistencies and improved compliance with quality management systems
Extensibility Implementation	Development of additional cookbooks for manufacturing and enterprise resource planning systems	Framework expansion beyond initial use cases demonstrating platform value for broader automation

Table 3: Phased Framework Deployment and Operational Results [7,8]

5. Discussion and Future Directions

The successful deployment and operational results from the case study validate the efficacy of integrated PowerShell and Chef automation for enterprise onboarding scenarios, while also illuminating several considerations for organizations contemplating similar automation initiatives. The substantial time savings achieved through automation derive not merely from replacing manual tasks with scripted equivalents, but from the elimination of sequential dependencies and wait times inherent in manual processes where technicians must coordinate across multiple systems and await availability of shared resources. The framework's parallel execution of identity management and application configuration supports this point because Chef client runs can occur concurrently with directory operations, without waiting for provisioning to finish, and can then deploy the application. Organizations that wish to take full advantage of automation should review their current onboarding processes to see if they can incorporate parallelization into their workflow and reduce unnecessary coordination delays that add up in a sequential process.

The framework illustrates key takeaways from finding the right balance between standardizing and providing flexibility in enterprise automation. The general aim is to standardize onboarding processes to minimize variability and to eliminate errors; however, real-world enterprise environments will always require exceptions and configurations with limited deviations for business or operational purposes. The framework overcomes this issue, through the use of parameterized PowerShell scripts, as well as Chef recipes that accept configuration variables to define behavior specific to the environment, allowing the same code base to run many deployment scenarios without unique scripts or cookbooks for each configuration. However, excessive flexibility introduces complexity that can undermine automation reliability, as proliferation of conditional logic branches increases testing burden and creates opportunities for edge cases that manifest as runtime failures. Organizations must establish governance processes that balance standardization mandates with legitimate business requirements for specialized configurations, potentially implementing approval workflows for

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

exceptions that deviate from standard automation patterns and trigger manual review of proposed configurations.

The technical architecture demonstrates several design patterns that contribute to framework robustness and operational reliability. The separation of orchestration, execution, and validation concerns enables independent evolution of each layer, allowing updates to Chef cookbooks without requiring modifications to PowerShell identity management scripts or orchestration logic. Idempotency principles that have been implemented throughout the framework ensure that if automation workflows are re-executed due to a transient failure, duplicate resources or conflicting states are not created. This is especially important for production automation systems encountering transient network faults, system upkeep, or resource contention to cause degradation for workflow automation systems. End-to-end logging and tracking the required state add sufficient operational visibility to give IT teams the ability, if an outage were to occur, to investigate operational concerns, systematically inquire whether there are recurring inconsistency patterns, and improve automation scripts based upon operationally-sourced empirical data.

Security considerations warrant particular attention in enterprise automation frameworks that manage identity lifecycle and application access. Artificial intelligence and machine learning integration in DevOps environments enables intelligent automation through predictive analytics for capacity planning, anomaly detection in system performance metrics, automated root cause analysis of infrastructure failures, optimization of resource allocation across cloud and on-premises environments, and enhanced security monitoring through behavioral analysis of user activities and system events [9]. The framework implementation incorporated role-based access controls for workflow initiation, mandatory code review for cookbook modifications, and comprehensive audit logging, but these safeguards required deliberate design decisions rather than emerging automatically from tool capabilities.

Future research directions for enterprise onboarding automation span several complementary areas. Integration with artificial intelligence and machine learning techniques could enable predictive onboarding that automatically provisions access to applications and resources based on analysis of similar employees' usage patterns. Natural language processing of onboarding requests could extract required parameters from free-text manager submissions rather than requiring structured form inputs. Extension of automation principles beyond initial onboarding to encompass ongoing access lifecycle management would provide comprehensive identity and access management automation spanning the entire employee lifecycle.

The convergence of DevOps practices with traditional enterprise IT operations represents a broader transformation that extends far beyond the specific onboarding automation scenario examined in this paper. DevOps performance measurement focuses on key metrics, including deployment frequency, measuring how often organizations successfully release to production, lead time for changes, tracking the duration from code commit to production deployment, mean time to recovery, assessing how quickly services are restored after incidents, and change failure rate, quantifying the percentage of deployments causing production failures requiring remediation [10]. The framework presented here demonstrates practical pathways for modernizing enterprise IT operations through incremental automation adoption that delivers measurable business value while respecting constraints of legacy application ecosystems and existing infrastructure investments.

Journal of Information Systems Engineering and Management
2026, 11(2s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Research Direction	Enabling Technologies	Expected Capabilities
Predictive Onboarding	Artificial intelligence and machine learning techniques	Automatic access provisioning based on analysis of employee usage patterns
Natural Language Processing	Text extraction and parameter parsing systems	Automated extraction of required parameters from free-text manager submissions
Lifecycle Management Extension	Comprehensive identity and access management automation	Coverage spanning role changes, project transitions, and offboarding procedures
DevOps Performance Optimization	Key metrics including deployment frequency and mean time to recovery	Enhanced operational efficiency through incremental automation adoption and measurable business value

Table 4: Future Directions for Enterprise Onboarding Automation [9, 10]

Conclusion

The automation framework discussed shows that combining PowerShell scripting with Chef configuration management can profoundly enhance enterprise onboarding processes for Product Lifecycle Management environments. The three-layer architectural structure of orchestration, execution, and validation allows organizations to remove manual coordination barriers, minimize configuration mistakes, and provide a consistent onboarding experience for distributed operations around the world. Successful delivery at a global aerospace manufacturing enterprise demonstrates the framework's ability to handle large volumes of onboarding requests without sacrificing success rates and realizing significant operational efficiencies. The framework's design highlights important architectural properties, such as concerns that were separated across discrete layers, idempotency to enable reliable re-execution capabilities, comprehensive logs for operational visibility, and secure credential management through privileged entitlement management. Organizations embracing this framework will be able to go beyond simply adopting automation for onboarding scenarios and automate enterprise identity and access management across the employee lifecycle, including role changes, project changes, and offboarding. The intersection of DevOps best practices with enterprise IT operations will change the way organizations deliver IT operations and services. Infrastructure-as-Code approaches will provide a foundation for ongoing modernization that creates real business value, while mitigating existing technology challenges. Future directions include integration of artificial intelligence and machine learning techniques for predictive onboarding, natural language processing for request parameter extraction, and automated optimization of configuration logic based on execution telemetry, advancing toward increasingly intelligent and adaptive enterprise automation platforms that continuously improve operational efficiency and service delivery quality.

References

[1] Michael Buckbee, "Active Directory Domain Services (AD DS): Overview and Functions," Varonis 2022. [Online]. Available: https://www.varonis.com/blog/active-directory-domain-services

[2] AltDigital Technologies, "What is Chef Infra?". [Online]. Available: https://www.altdigital.tech/resources/altdigitalpedia/chef-infra

[3] David Sandilands and Margaret Lee, "2024 State of DevOps Report: Platform Engineering Edition," Puppet, 2024. [Online]. Available: https://www.puppet.com/blog/state-devops-report-2024

[4] Sulaiman Asif, "Chef Architecture: Overview of Chef Infra," Upgrade KnowledgeHut, 2025. [Online]. Available: https://www.knowledgehut.com/blog/devops/chef-architecture

[5] Microsoft Ignite, "Security for SQL Server Database Engine and Azure SQL Database," 2025. [Online]. Available: https://learn.microsoft.com/en-us/sql/relational-databases/security/security-center-for-sql-server-database-engine-and-azure-sql-database?view=sql-server-ver17

[6] CoConote, "Introduction to InSpec Compliance Automation," 2025. [Online]. Available: https://coconote.app/notes/f9e0f480-b3c5-4ec6-925a-c8ad196bb5dd

[7] Pluralsight Content Team, "How do I upskill into a DevOps role?" Pluralsight, 2023. [Online]. Available: https://www.pluralsight.com/resources/blog/business-and-leadership/how-to-upskill-learn-devops

[8] FORTNA, "5 Steps to Building a Business Case for Automation." [Online]. Available: https://www.fortna.com/insights-resources/5-steps-to-building-a-business-case-for-automation/

[9] Cogent Info, "AI and Machine Learning in DevOps," 2024. [Online]. Available: https://www.cogentinfo.com/resources/ai-and-machine-learning-in-devops

[10] Salesforce, "5 Important DevOps Metrics and KPIs to Measure." [Online]. Available: https://www.salesforce.com/platform/devops-tools/what-is-devops/metrics/